Routing Through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments

Jalal Al-Muhtadi* Roy Campbell∇ Apu Kapadia† M. Dennis Mickunas∇ Seung Yi
Department of Computer Science,
University of Illinois at Urbana-Champaign,
{almuhtad, rhc, akapadia, mickunas, seungyi}@uiuc.edu

Abstract

Ubiquitous computing is poised to revolutionize the way we compute and interact with each other. However, unless privacy concerns are taken into account early in the design process, we will end up creating a very effective distributed surveillance system, which would be a dream come true for electronic stalkers and "big brothers." We present a protocol, which preserves the privacy of users and keeps their communication anonymous. In effect, we create a "mist" that conceals users from the system and other users. Yet, users will still be able to enjoy seamless interaction with services and other entities that wander within the ubiquitous computing environment.

Keywords

Ubiquitous computing, privacy, Mist Routers, anonymous communication, authentication, security

1. Introduction

The major advances in distributed systems and mobile computing have converged to enhance global interconnectivity. This has fueled the idea of ubiquitous computing and active information spaces where users can access services, run programs, utilize resources, and harvest computing power anytime and anywhere. This new generation of ubiquitous computing enables the delivery of integrated services and multimedia-enabled applications that are no longer bound by time or location barriers. Ubiquitous computing promotes the proliferation of embedded devices, smart gadgets, sensors and actuators. These devices will be everywhere, performing regular tasks, providing new functionality, extending the reach of traditional computing to physical spaces, and allowing users to interact seamlessly with the surrounding environment.

Physical spaces augmented with sensors and actuators that can locate users, detect their presence, and track their whereabouts will be commonplace in this new and exciting computing paradigm. These sensors will play a major role in bridging the virtual computing world with the physical world and boosting the productivity of users and the availability of computing resources. However, these very features could severely threaten the privacy of users. For instance, the mentioned services can be exploited by intruders, malicious insiders, or even "curious" system administrators to track or electronically stalk particular users. Although encryption provides confidentiality by hiding information flowing through communication channels from eavesdroppers (e.g., an insider or a system administrator), an eavesdropper can still gather the network addresses or physical locations of the communicating parties. The lack of privacy in today's networks and distributed systems is well-documented [10][16]. Similar concerns arise for ubiquitous computing environments [8]. While several approaches have tried to address these problems (see Section 2) the solutions presented are either only concerned with anonymous web browsing or with trusted third parties that store the location information of users and only disclose it to authorized principals.

In this paper we aim to design and implement a privacy protocol that allows users of a ubiquitous computing environment to roam and communicate freely while preserving their privacy. The privacy protocol prevents insiders, system administrators and even the system itself from tracking users and detecting their physical location. Yet, the system will enable users to communicate with other users and access computing resources in an authenticated manner without disclosing the users' physical locations or whereabouts. Further, users will be able to configure the level of privacy they wish to enjoy through the use of a user interface running on their mobile devices (e.g., mobile phone, laptop, PDA). We plan to achieve this by allowing the ubiquitous computing environment to maintain

* Jalal Al-Muhtadi is funded by a grant from the National Science Foundation, NSF CCR 0086094 ITR.

∇ These authors are supported by grants from the National Science Foundation, NSF CCR 00-86094 ITR, NSF EIA 98-70736, NSF EIA 99-72884 EQ, and NSF CCR 00-86094.

† Apu Kapadia is funded by the Department of Energy High Performance Computer Science Fellowship through Los Alamos National Laboratory, Lawrence Livermore National Laboratory, and Sandia National Laboratories.

sensors that can detect the presence of users in a room, but without the ability to positively identify the users. Combined with our routing protocol, this creates a "mist" through which users can communicate privately. In our system, we introduce an overlay network in the form of a hierarchy of "*Mist Routers*" that perform "*handle-based routing*" to preserve privacy and hide information about the original source and the final destination. In short, we refer to this hierarchy as a "*Mist Hierarchy.*" The handle-based routing combines hop-to-hop routing based on handles with limited public-key cryptography to preserve privacy from eavesdroppers and traffic analyzers. Positive authentication and registration of users can be achieved at a higher level in the hierarchy, making it harder to infer the user's current location.

Our privacy scheme is being deployed in Gaia. Gaia [6][11][12] is a component-based operating system built on a reflective middleware layer. Gaia is being developed at the University of Illinois at Urbana-Champaign to provide an infrastructure in the form of core services over which active information spaces can be constructed. Through Gaia, a plethora of platforms, and hence devices, can be "Gaia enabled" by running the Gaia OS middleware on top of the installed operating systems. The Gaia OS glues together all such devices and enables ubiquitous computing environments.

Assumptions

Privacy is a fuzzy term that is often overloaded to mean a large variety of things. Therefore, before proceeding any further, it is important to clarify the scope of user privacy that we strive to achieve in a ubiquitous computing environment. Our goal is to achieve the following:

1. *Location privacy:* Neither the system nor the users of the system will be able to know the exact physical location of a user, unless that user decides to disclose such information or if another person physically sees that user at that location.
2. *Anonymous connections:* If two parties decide to communicate with each other, then other users in the system will not know who the communicating parties are, unless one of the communicating endpoints decides to disclose such information.
3. *Confidentiality:* If both endpoints of a communication agree, they can make the content of their communication confidential, such that neither the system nor other users in the system can read the contents of the communication.

We assume that a Public Key Infrastructure (PKI) exists for users of the ubiquitous computing environment and for the Mist Routers in the system. However, we do not assume the existence of a third party that can be trusted to safely store sensitive information about users, like their physical location for instance.

The remainder of this paper is divided as follows. Section 2 talks about related work. Section 3 describes the details of the proposed system. Section 4 shows our implementation. Section 5 gives pointers to future work. Finally, Section 6 concludes.

2. Related Work

In this section, we present some of the existing research that relate to our system. Compared to the amount of research efforts directed towards ubiquitous computing, very little attention has been paid to the security aspects of ubiquitous computing so far. However, in this section, we will consider some of the approaches that attempt to achieve anonymity on the Internet. Some projects try to provide a way to hide a user's identity while communicating over an open network while others try to provide a communication channel that is immune to traffic analysis, hence, providing anonymity from eavesdroppers. We describe some of the representative works in this section.

In [8], Marc Langheinrich warns us about the possibility of an Orwellian nightmare in which current ubiquitous computing research continues on without considering privacy protection in the system. He proceeds to describe the design principles of privacy-aware ubiquitous systems. Some of the principles proposed are yet to be implementable with current technology but the paper gives a good general guideline for privacy issues in ubiquitous computing systems. The crucial point of this paper is that unless you consider the privacy concerns since the initial stages of a ubiquitous system design, it is very likely to end up becoming a ubiquitous surveillance system. Our approach fits the spirit of this paper in the sense that we integrate the privacy concerns into the routing itself.

Previous research on privacy and anonymity on the Internet can be classified into roughly two categories: user anonymity and anonymous communication. User anonymity aims at providing the users anonymity while they are using the network by letting them hide their identity from the communicating peers. Research on anonymous communication focuses on providing a communication channel that is immune to traffic analysis so that the communicating parties can be anonymous against the eavesdroppers.

Anonymizer [1] and SafeWeb [15] are two user anonymity solutions provided to World Wide Web users. Anonymizer is a centralized approach to hide the web users' real identities from the web servers they access. Users can enjoy anonymity by rerouting their HTTP packets through the Anonymizer, which replaces the information in the packet headers so that the websites cannot infer the users' identities. This approach has the problem of a centralized trusted entity. The Anonymizer site can track all

the anonymous user activities and is also a single point of failure.

Crowds [10] by Aviel Rubin et al. is one of the approaches on anonymous communication. A Crowd is a set of voluntarily cooperating hosts. Any message that requires anonymity first channels into one of the Crowds hosts and then enters a loop until it finally gets out of the Crowd and arrives at the destination. Using statistical forwarding decisions, Crowds can effectively hide the communication pattern of a user. Another similar approach is Onion Routing [9]. Users can use the deployed set of Onion routers in the Internet to achieve a level of privacy similar to that of Crowds. One difference, however, is that the Onion routers themselves form a ring and keep constant TCP connections between the neighboring routers, constantly transmitting packets through the routes. Also, packets are encrypted with multiple keys to form an "onion," so none of the Onion routers forwarding the packets can discover both the source and the destination information of the packet. NetCamo [7] is an approach to counter traffic analysis in real-time. NetCamo models the traffic patterns of nodes or networks and provide a real time rerouting and padding to hide the communication pattern.

All of these approaches do not authenticate users before allowing them to join or use the service. Anyone can use the service without revealing anything about himself or herself, which can become another problem when exploited by malicious users. Further, if a service requires user authentication, then this may threaten user anonymity, rendering the privacy protocols useless. In our system, however, users need to authenticate themselves first in the registration process in such a way that would neither threaten their communication anonymity nor would reveal their physical location, so we do not have such a problem.

Zero Knowledge Systems' Freedom.net product [5] provides a bit of both directions. First, it uses Freedom servers scattered over the Internet just like in Crowds to hide their communication patterns. Also, Freedom.net users can install a client program in their end host and use a cryptographically protected pseudonym called 'nym' to connect to the Internet, doing telnet, ftp, web surfing and emailing. Freedom.net users need to authenticate themselves to the system before using the service[1]. Our approach achieves many of the features provided by Freedom.net service. Moreover, our Mist provides additional level of privacy by not revealing the user's location even to his or her communicating peer.

[1] Unfortunately, at the time of this writing, Zero Knowledge Systems stopped providing the Freedom.net service. This may show the market's indifference towards privacy on the Internet, but we believe that privacy will be one of the key issues in accepting and deploying ubiquitous computing technologies.

Another related research project is the Cricket location support system [14]. In Cricket, beacons are deployed in the active information spaces to provide users with location information. Cricket uses RF and ultrasound to provide an accurate estimation of user's location. Since users just have to listen to the beacon messages to determine their location, users do not reveal any information about themselves to the environment, hence maintaining their privacy. Although Cricket does not require users to reveal their identity, there are limitations on the actions users can do with the acquired location information if users do not transmit anything to the environment. Mist provides an authentication process so that users can authenticate themselves to the system and safely interact while still preserving their privacy.

3. System Design

In our system, Mist Routers are deployed in a hierarchical fashion. Users connect directly to one of the leaf level Mist Routers, which we call "Portals." Through a Portal, a user (or the user's device) sets up a "Mist Circuit" upwards in the hierarchy. A Mist Circuit is a handle-based virtual circuit between the user and a special Mist Router, which we call a "Lighthouse." Since the handles for the virtual circuit is set up on a hop-by-hop basis, unless enough Mist Routers in the path collude, none of the intermediate Mist Routers can deduce the two ends of the virtual circuit. A user uses Mist Circuits to contact one of the higher level Mist Routers who is willing to serve as a contact point for that user. This contact point will only have partial information on how to route to that user. We refer to this contact point as a "Mist Lighthouse" for that user.

3.1 Mist Hierarchies

Mist Routers are key elements in our system. They conceal the identity and location of communicating parties by rerouting packets among themselves using hop-to-hop handle-based routing (which we describe in more details in Section 3.2). We envision that Mist Routers will be deployed in hierarchical clusters organized along physical space divisions, called domains. The hierarchical organization of Mist Routers would enhance the system's flexibility and scalability, allowing it to be easily deployed over multiple domains.

Initially, a Mist Hierarchy needs to be agreed upon and constructed between the different physical space domains that are willing to cooperate and provide privacy for users roaming in them. Meeting this requirement should not be a problem; this is because many physical spaces are organized into hierarchies by nature (e.g. as illustrated in Figure 1). Further, such hierarchies can be constructed dynamically in a fashion similar to the way multicast protocols construct source-based trees [4] or Core Based Trees (CBT), which builds a "shared tree" rooted at a core router

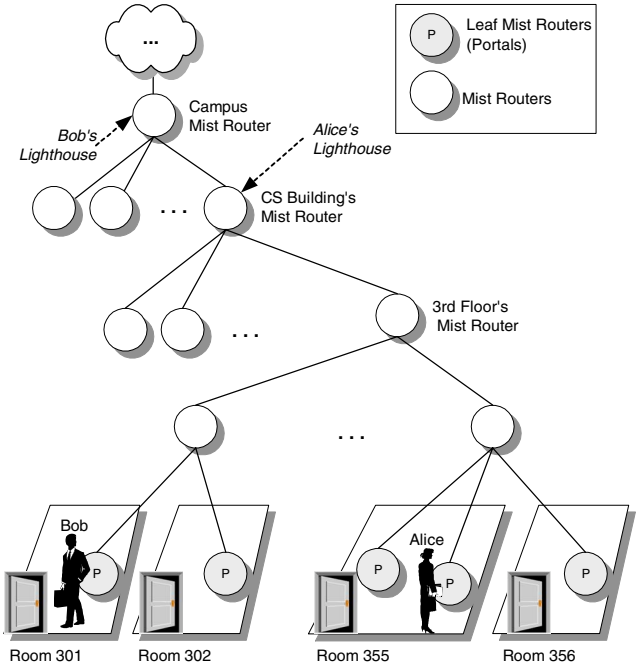


Figure 1: The Mist Hierarchy

[3]. Allowing these hierarchies to spread over multiple domains make it harder for corrupt Mist Routers to collude.

As illustrated in Figure 1, Mist Routers at the leaves of the hierarchy represent "Portals." Portals are viewed as the gateways that bridge the virtual world to the physical one. In other words, they are connection points where users of an active information space can connect to the system. Portals are represented by a variety of hardware that can include a fixed workstation, a sensor, an access point for wireless devices, and an RF transceiver.

The Portals in our system will be able to detect the presence of users in a room, but without the ability to positively identify them. In other words, the "smart" rooms will be able to detect the physical presence of one or more users. However, as far as the smart room and its Portals are concerned, the users are anonymous and not authenticated as of yet. In this paper, we rely on existing discovery and location detection protocols to sense the existence of users in the room, like the location and discovery services that are available in Gaia OS [12]. We also assume that the spaces supporting our privacy system would not contain surveillance cameras or voice recognition devices, otherwise, users will have to take additional physical precautions to protect their privacy, like wearing masks or staying silent!

As previously indicated, the original objective of an active information space is to allow seamless interactions between the various virtual and physical entities in the space. Therefore, there should be a mechanism over which these interactions can take place in spite of the existence of this mist that blurs the true identities of users and hide their physical locations. Therefore, to access the system, to communicate with others, and to use available resources while maintaining privacy, user Alice, say, has to register herself in the system as shown in Figure 2. The registration takes place through Alice's mobile device (which can be a PDA, a mobile phone, or even a smart badge). The device talks directly to one of the available Portals in the surrounding physical space. The mechanism involves designating a special Mist Router for every user of the system. This special Mist Router will be referred to as a "Lighthouse" for that user. For example, a Lighthouse for Alice is a Mist Router that is an ancestor of the Portal that Alice is connecting to. Alice's Lighthouse will have knowledge of her true identity as well as partial knowledge on how to route to Alice. However, it does not know the exact physical location of Alice. Whereas the Portal knows the exact physical location of Alice, but does not "realize" that this is actually Alice and does not know who Alice's Lighthouse is. Going back to the registration process illustrated in Figure 2, Alice's device sends a registration request to the nearby Portal. The Portal will reply back with a list of its ancestral Mist Routers that exist at a higher level within the Mist Hierarchy and are willing to act as a Lighthouse for the user. A trusted third party can be used to vouch for the trustworthiness of some of these Mist Routers, particularly the ones that exist near the root of the hierarchy, since these Mist Routers can be accessible from different spaces. This vouching process is similar to how certificate authorities vouch for other parties on the Internet. User Alice, through her PDA device, can customize the amount of privacy she wishes to enjoy by selecting a Mist Router at a suitable height in the hierarchy to be her Lighthouse. Selecting a Lighthouse is a tradeoff between performance and privacy. Choosing a Mist Router that is closer to the root of the hierarchy provides better privacy because less information is inferred about the actual physical location of Alice, and the extra rerouting provides better concealment. Whereas selecting Mist Routers closer to the Portal helps performance by limiting the number of reroutes but decreasing the level of privacy. To illustrate, in Figure 1, Alice decides to designate the Computer Science building's Mist Router as her Lighthouse. This information implies that Alice is currently located somewhere in the Computer Science building. Bob, on the other hand, chooses the campus Mist Router as his Lighthouse. This

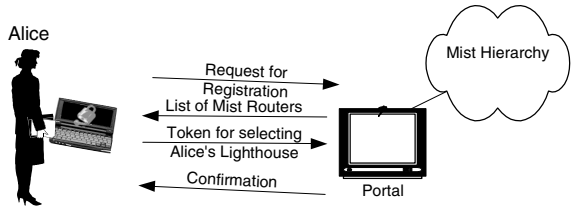


Figure 2: Registering in the system

implies that he physically can be anywhere in campus. Ultimate privacy can be achieved when a user chooses the hierarchy's root as its Lighthouse.

Upon the selection of a suitable Lighthouse by Alice, we establish what we refer to as a "Mist Circuit" between Alice and the selected Mist Router. We discuss Mist Circuits in more detail in the Section 3.2. In any case, the Mist Circuit will make it possible for Alice's Lighthouse to authenticate Alice while hiding her exact physical location, and, at the same time, hiding her identity and her selected Lighthouse from the Portal she is connected to.

We note here that if Alice is a highly-mobile user moving from one room to another while communicating, then prompting Alice repeatedly about selecting a Lighthouse goes against the original goals of ubiquitous computing. To solve this problem, Alice's mobile device can be configured to automatically "remember" the Mist Router that Alice selected as her Lighthouse. The device can then perform the registration process transparently without Alice's intervention. However, when Alice moves into an area where the selected Lighthouse can no longer be accessed, only then Alice is warned and prompted to select another Lighthouse. A prioritized list of "preferred Lighthouses" can be stored in Alice's mobile device, allowing Mist registration to take place transparently.

3.2 Mist Circuits

Mist Circuits employ hop-to-hop, handle-based routing to send data packets back and forth between the source and destination through the mist. Combining this routing with limited public-key encryption allows data packets to be successfully routed through the mist while providing a higher degree of privacy and concealment. This prevents intermediate nodes from recognizing the identities of the actual endpoints or their physical location. Recall that we establish a Mist Circuit between the user and its selected Lighthouse so that the user can reveal its true identity and authenticate it at the Lighthouse without disclosing physical location information. In this section we describe how a Mist Circuit is set up and used.

We go back to the example of Alice registering in the system. Her Portal fulfills her request for registration by replying back with a list of ancestral Mist Routers that are willing to act as Lighthouses. The list returned contains two pieces of information for each Mist Router. Each entry will contain an ID that uniquely identifies the Mist Router and a digital certificate for that Mist Router. The digital certificate can be issued by some trusted third party. The certificate could contain information about the how "high" in the Mist Hierarchy the associated Mist Router is. In other words, the list is of the form:

$<Mist\ Router_1, Certificate_1>,$
$<Mist\ Router_2, Certificate_2>,$
…

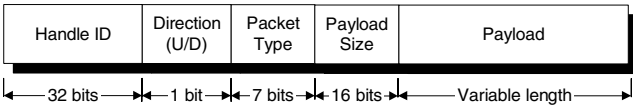


Figure 3: General format for Mist packets

User Alice selects a suitable Mist Router, which she does not disclose to the Portal. To establish a Mist Circuit, Alice generates a Mist Circuit establishment packet. The general format of Mist packets are illustrated in Figure 3. The 'Handle ID' field represents a handle that is unique per Mist Router that helps identify the next hop on the packet's route. A value of 0 in this field indicates that no value is assigned yet. How the handle is used is described later in this section. The 'direction' field is a single bit that specifies whether the packet is going upwards (toward the Lighthouse) or downwards (toward the Portal) in the hierarchy. The 'packet type' identifies the type of the packet, which tells the intermediate Mist Routers how they should handle the packet.

Assuming that Alice selects the Mist Router 'Z' in Figure 4 as her Lighthouse, then Alice's Mist Circuit establishment packet will contain '0' for the handle ID and 'U' in the direction field, indicating that this packet is going upwards. The type field will contain a value indicating that this is a Mist Circuit establishment packet. The payload will consist of the Message *M*:

$M = E_{public\_key\_Z}(Alice \,||\, TS \,||\, K_{session} \,||\, TKN \,||\, PP)$

Where:

|| stands for concatenation.

Alice: Alice's unique ID in the active information space

TS: A timestamp to prevent replay attacks.

$K_{session}$: A random session key to encrypt further communication between the user and her or his Lighthouse. It is also used to add some additional randomness into the encrypted message.

TKN: A token to be presented to the user's lookup service. Details about the user's lookup service and the contents of this token are given in Section 3.3.

E_k: Means encrypt using the key '*k*'.

PP: A predetermined "fixed" phrase. In our current implementation, we are using the string "Mist Circuit Establishment Message." The use of this will be described below.

The actual payload is:

$Payload = M \,||\, S_{Alice}(M)$,

where $S_{Alice}(M)$ indicates Alice's digital signature over M.

The contents of the Mist Circuit establishment packet are shown in Figure 5. Alice then transmits this packet to her Portal, without informing the Portal of the selected Lighthouse. Portals will maintain a table that is referred to

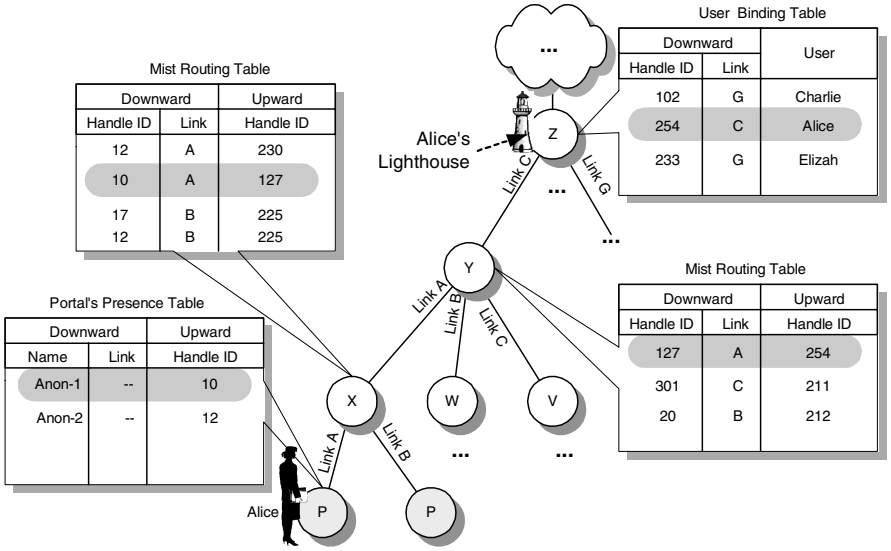


Figure 4: Mist Circuit setup

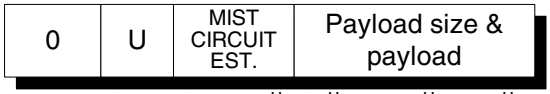
as the "Presence Table." Since the Portal detects nearby people without positively identifying them, whenever a new person is detected, he or she is entered into the Portal's presence table as an "anonymous" person. Additionally, the Portal assigns for every user a handle ID that is unique within that table only. So in the scenario depicted in Figure 4, Alice is represented as "Anon-1" and is assigned a handle ID of 10, say. If other users exist in the same physical space and the Portal is able to communicate with them, then similarly, they will be entered into the presence table. The "link" field should contain a value that identifies the network link or port number over which the Portal can communicate with the corresponding user. We assume that if a Portal supports communication with more than one physically present user, then it should be able to recognize which user sent a particular packet. Upon receipt of the Mist Circuit establishment packet from Alice, the Portal will replace the value in the packet's handle ID field with the handle ID that was assigned to Alice in the presence table, which is 10 in the example shown. Next the Portal will transmit the modified packet "upward" to its parent Mist Router.

From now on, upon receiving the circuit establishment packet every intermediate Mist Router will attempt to decrypt the encrypted portion of the payload using its private key. If the decryption fails, (the predetermined phrase can be used to indicate whether or not the decryption failed) then the Mist Router will infer that this packet is not meant for it. Instead, the packet has to be passed upward to its parent. Each Mist Router will maintain a "Mist Routing Table." This table will associate handle IDs used over downward connections with handle IDs that will be used on the upward connection. Note that within the downward column of the Mist Routing Table, the combination of Handle ID and link ID is unique per Mist Router, whereas, within the upward column the handle ID value is unique per Mist Router. The current Mist Router does a quick lookup on its Mist Routing Table to see if it has an entry for the handle ID and the link over which it received the packet. If it does not, it creates one, and associates an upward handle ID for it. The Mist Router then substitutes the value of the packet's handle ID with the newly assigned value and passes the message to its parent. The process is repeated for every intermediate Mist Router.

On the other hand, if a Mist Router successfully decrypts the encrypted portion using its private key, then this indicates that the user actually chose the current Mist Router as his or her Lighthouse. All Mist Routers that are willing to act as Lighthouses for users should maintain a 'User Binding Table' as shown in Figure 4. The Mist Router can now authenticate the user by verifying his or her signature and checking the freshness of the timestamp. The handle ID and the downward link above which it was used will be stored in the User Binding Table, along with the actual ID of the user.

Figure 4 shows the actual entries in the presence, routing and binding tables when user Alice registers and chooses 'Z' as her Lighthouse. The shaded entries in the figure represent Alice's entries. In effect, this process has established a "circuit" over which Alice can communicate with her Lighthouse securely. Note that while Alice's Lighthouse can infer that Alice exists somewhere in the hierarchy underneath Mist Router 'Y', the exact location cannot be determined unless enough Mist Routers agree to cooperate. Therefore, the longer the path between Alice and her Lighthouse the more "private" her location becomes.

To complete the Mist Circuit establishment, the Lighthouse confirms the registration of Alice by sending back a reply packet. The format of this reply is shown in Figure 6. For the example shown, the handle ID will be set to 254, because this is the value bound to Alice. The packet should be sent downward (D). The packet type is set to



$$M = E_{\text{public key of Z}}(Alice \,||\, TS \,||\, K_{session} \,||\, TKN \,||\, PP)$$

$$Payload = M \,||\, S_{Alice}(M)$$

Figure 5: Alice's Mist Circuit establishment packet

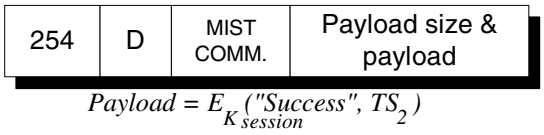


Figure 6: Registration confirmation packet

"MIST COMMUNICATION" which indicates that intermediate Mist Routers should not attempt to decrypt the contents, rather, they should just route it to the next hop.

$K_{session}$ is the session key between the Mist Router 'Z' and Alice that was transmitted through the Mist Circuit establishment packet. Note that to improve performance from this point on, we use symmetric encryption to achieve confidentiality between the user and the chosen Lighthouse. TS_2 is a timestamp to prevent replays. This packet can now be routed back to Alice in a manner similar to what was described above. Now Alice can communicate securely with her Lighthouse while preserving her privacy.

Note that if an intermediate Mist Router goes down; its subtree will be disconnected from the rest of the Mist Hierarchy. Since Mist Routers form an overlay network over the conventional network, this failure does not physically partition the network, and the Mist Hierarchy can be reestablished. For example, the children of the failed Mist Router can be connected to its parent. We are currently investigating such algorithms for actively maintaining the Mist Hierarchy, and increasing its resilience to such failures.

3.3 Locating Users

Once the Mist Circuit-Setup has been completed, the Lighthouse Mist Router acts on behalf of the end-user. All communication with the user will take place through its Lighthouse, since only the Lighthouse knows how to route packets to the user. However, we first need to locate the current Lighthouse for a particular user. Only then can one communicate with the user. We present two approaches that would be suitable for performing lookups that return the location of the current Lighthouse based on the user's name. Each approach involves the *registration* of <user, Lighthouse> pairs, and the *lookup* of <user, Lighthouse> pairs.

3.3.1 LDAP Servers

RFC 1777 describes the Lightweight Directory Access Protocol (LDAP). In essence, users can register attributes with LDAP servers, which can consequently be looked up with a subset of these attributes. Mist users will have a unique LDAP *Distinguished Name (DN)*. Mist users can look up information about other Mist users either based on their DN's, or on their attributes. For example, one could look up a user based on the last name and university, "Doe from University of Illinois." Once a user has been located, the attribute corresponding to the current Lighthouse can be retrieved.

3.3.2 Web Servers

Another interesting technique would be to allow users to maintain their own webpages. These webpages can be updated by a CGI script, for example, to contain the current Lighthouse's location. Every time a Mist user registers with a Lighthouse, the Lighthouse will update the user's webpage via the CGI interface with its identity. Now other Mist users can simply lookup the user's webpage for the current Lighthouse's location. In such a scheme other Mist users will have to be aware of the other users' webpage URLs.

3.3.3 Security issues

We would like to prevent malicious Lighthouses or attackers from falsely registering users with them. To achieve this, the user constructs a special token (*TKN*) signed by the user's private key. This token will contain a timestamp and the unique ID of the chosen Lighthouse. This token is propagated to the Lighthouse during the Mist Circuit setup as described in Section 3.2. Once the Mist Circuit has been established, the Lighthouse presents this token to the lookup service. For example, this can be presented to the LDAP server, or to the CGI script. In both cases, these updates will be secure, and cannot be forged or replayed by an attacker. If the timestamp has already been seen before, or if it has expired, the token will be discarded. Naturally, if the signature cannot be verified, the token is also discarded. The format of this token, *TKN*, is as follows:

$$TKN = (User\ ID \,||\, Lighthouse\ ID \,||\, Timestamp \,||\, S_{User}(User\ ID \,||\, Lighthouse\ ID \,||\, timestamp))$$

This tells us that *TKN* contains the user ID (for example, in LDAP we would use the user's DN), the Lighthouse ID (this could be the DNS name) and the timestamp are signed by the user's private key. *TKN* contents do not need to be encrypted because the contents are already known by the Lighthouse anyway. Hence, only integrity of this message, not confidentiality, needs to be guaranteed.

3.4 Mist Communication Setup

Once we have located the Lighthouse for a particular user, we need to set up a communication channel through it. In our system we assume that both users in the communication setup have established their own Mist Circuits and are both registered with their respective Lighthouses. Communication will now take place through the two Lighthouses. We will use the notation $Lighthouse_X$ to mean "Lighthouse of User X." Let us say that Bob is trying to initiate communication with Alice. Bob and Alice are registered with $Lighthouse_{Bob}$ and $Lighthouse_{Alice}$ respectively.

Bob generates the following message for its Lighthouse:

$M_{Lighthouse} = E_{Ksession}(COMM\_SETUP \,||\, Alice's\ ID\ or\ attributes \,||\, TS)$

Note that all messages in this section are actually the payload of Mist Communication packets. Since handles have been set up in both directions during the Mist Circuit Setup phase, this message will travel up to $Lighthouse_{Bob}$. Note that intermediate Mist Routers are never aware of the user's Lighthouse. When the message arrives at $Lighthouse_{Bob}$ it is able to uniquely determine that the message is from Bob based on the arriving handle. It decrypts the message with session key $K_{Session}$ and determines from the *COMM_SETUP* message type that communication must be set up with Alice. If Alice's ID is included then the lookup for $Lighthouse_{Alice}$ is straightforward. However, if Bob specifies attributes, then $Lighthouse_{Bob}$ must perform a lookup based on these attributes. If a unique match for Alice is found based on these attributes, $Lighthouse_{Bob}$ can determine Alice's ID. In both cases, Alice's ID is used to lookup $Lighthouse_{Alice}$. The timestamp *TS* is used to prevent replay attacks.

$Lighthouse_{Bob}$ uses asymmetric key encryption with $Lighthouse_{Alice}$ to determine $Lighthouse_{Alice}$'s handle for Alice. Since this is straightforward, we avoid the details of this communication. We will call this the destination handle for Alice, or $dest\_handle_{Alice}$. In Figure 7, we can see that $Lighthouse_{Bob}$ determines $dest\_handle_{Alice}$ = 254-C. $Lighthouse_{Bob}$ then generates a unique handle that Bob can use to address Alice. We will call this handle src_handle_{Alice}. In Figure 7 src_handle_{Alice} = 689. $Lighthouse_{Bob}$ sets up a binding of the form <src_handle_{Alice}, $dest\_handle_{Alice}$, $Lighthouse_{Alice}$>. In Figure 7 we can see the binding <689, 254-C, Y>. We call this a Mist Communication Binding. All messages from Bob that arrive for src_handle_{Alice} (689) will be tunneled to $Lighthouse_{Alice}$ (Y) and indexed with $dest\_handle_{Alice}$ (254-C). Similarly, $Lighthouse_{Bob}$ will supply the handle for Bob to $Lighthouse_{Alice}$ that will set up a binding of the form <src_handle_{Bob}, $dest\_handle_{Bob}$, $Lighthouse_{Bob}$> in the same way. In Figure 7 we can see this binding as <412, 100-A, X>.

Once $Lighthouse_{Bob}$ and $Lighthouse_{Alice}$ have setup their bindings, they need to inform Bob and Alice of the src_handles. $Lighthouse_{Bob}$ sends src_handle_{Alice} to Bob in the following message:

$M_{Handle} = E_{Ksession}(HANDLE\_MSG \,||\, Alice's\ ID \,||\, src\_handle_{Alice} \,||\, TS)$

In Figure 7 this message corresponds to "For Alice use 689." Similarly, $Lighthouse_{Alice}$ sends src_handle_{Bob} to Alice.

Now Bob can send $Lighthouse_{Bob}$ messages destined to Alice by simply using src_handle_{Alice} (689), and Alice can send $Lighthouse_{Alice}$ messages destined for Bob using src_handle_{Bob} (412). This is done to hide Alice's identity from intermediate routers. These intermediate routers are hence unaware of *both* the endpoints of the communication. To communicate with Alice, Bob constructs messages of the following form, where 'M' is the message for Alice:

$M_{For\_Alice} = (COMMUNICATION\_MSG \,||\, src\_handle_{Alice} \,||\, M)$

This message will propagate upstream until it reaches $Lighthouse_{Bob}$, which uses src_handle_{Alice} (689) to determine $Lighthouse_{Alice}$ (Y) and $dest\_handle_{Alice}$ (254-C). Note that the Message passes in the clear, and the use of handles does not disclose the endpoints of the communication. Alice and Bob are now free to choose an end-to-end encryption scheme if desired. Using this method, there is no duplication of encryption by the Mist. Once $Lighthouse_{Alice}$ is determined, the Message M needs to be forwarded to $Lighthouse_{Alice}$. $Lighthouse_{Bob}$ sends the following message to $Lighthouse_{Alice}$. We use the subscript of "crossing" to suggest that the message is crossing over from one Lighthouse to another.

$M_{Crossing} = (dest\_handle_{Alice}, M)$, e.g., (254-C, M)

When $Lighthouse_{Alice}$ receives this message, it uses this $dest\_handle_{Alice}$ to route message M to Alice. Similarly, $Lighthouse_{Alice}$ can route messages to Bob using:

$M_{Crossing} = (dest\_handle_{Bob}, M)$, e.g., (100-A, M)

Note that these "crossing" messages between Light-

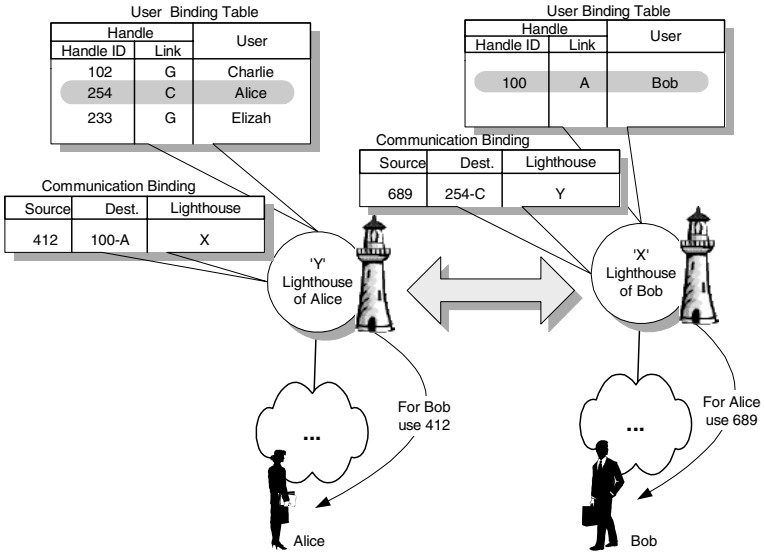


Figure 7: Mist communication setup

houses are not the Mist communication messages described before. The Lighthouses use their own packet formats to exchange the crossing messages.

3.5 Security issues

Here we discuss how the described scheme achieves location privacy for Alice and Bob. We also present an enhancement that adds anonymity to Alice and Bob's connection.

3.5.1 Privacy

Note that $Lighthouse_{Bob}$ and $Lighthouse_{Alice}$ are aware of the identities of the endpoints of the communication, but they are not aware of Alice and Bob's locations. Hence the privacy of Alice and Bob is preserved. In addition, all intermediate routers are unaware of the endpoints of the communication, and hence cannot deduce the locations of Alice and Bob. In fact Alice and Bob can communicate anonymously with respect to all other routers, with the exception of the two Lighthouses. With respect to this communication, the Lighthouses are trusted entities, and hence fully anonymous connections are not provided. In the next section we describe how fully anonymous connections can be achieved. In what we have described so far, we achieve our goal of preserving Alice and Bob's location privacy from all intermediate routers, including the Lighthouses. The most important thing to note is that Alice cannot deduce Bob's location, and Bob cannot deduce Alice's location. Hence communication between Alice and Bob is privacy preserving.

It is worth mentioning that if the user's Portal colludes with the user's Lighthouse, then the location and identity of the user may be compromised. However, note that the user's Portal does not know which Lighthouse the user selected (and vice versa). Hence, for this to be practical, the Portal has to employ a trial-and-error approach to try to get accessible Lighthouses to collude and help in decrypting the initial packet that was received by the Portal from the user. Our system distributes the trust, and assumes that such Lighthouses and Portals span various domains, and collusion between such entities is not feasible.

3.5.2 Anonymous connections

As noted in the previous section, even though Alice and Bob have achieved location privacy, their connection is not anonymous to the Lighthouses. Ideally, we would have a situation where $Lighthouse_{Bob}$ does not know that Bob is exchanging messages with Alice, and where $Lighthouse_{Alice}$ does not know that Alice is exchanging messages with Bob. In such a case Alice and Bob can communicate with full location privacy and connection anonymity, and the only information available to $Lighthouse_{Bob}$ is that Bob is communicating with "somebody," and likewise $Lighthouse_{Alice}$ knows only that Alice is communicating with "somebody." We detail an enhancement to the protocol described above to achieve this.

We first modify the message $M_{Lighthouse}$ Bob sends to its Lighthouse to include an encrypted token with Alice's ID. We will call this token T_{Alice},which is encrypted with the lookup server's key (we assume this key to be well known). Hence $Lighthouse_{Bob}$ is not aware of the identity of Alice.

$$M_{Lighthouse} = E_{Ksession}\ (COMM\_SETUP \,||\, T_{Alice} \,||\, TS)$$

Next, $Lighthouse_{Bob}$ sends T_{Alice} to the lookup server, which decrypts Alice's ID and determines $Lighthouse_{Alice}$. The lookup server creates another token $T_{LighthouseAlice}$, encrypted with $Lighthouse_{Alice}$'s key, which contains Alice's identity. Now $Lighthouse_{Bob}$ uses this token instead of Alice's ID with $Lighthouse_{Alice}$ to determine $dest\_handle_{Alice}$. The rest proceeds as before. Hence Bob can route packets to Alice without $Lighthouse_{Bob}$ knowing that the packets are for Alice. Similarly, $Lighthouse_{Bob}$ will provide src_handle_{Bob} to $Lighthouse_{Alice}$, but without disclosing Bob's identity. Hence Alice has a reverse communication path in which $Lighthouse_{Alice}$ does not know that Alice is communicating with Bob. Even if there are repeated communication setups between Alice and Bob, the token (e.g., T_{Alice}) will differ each time due to the inclusion of a timestamp. We mention this to emphasize that $Lighthouse_{Bob}$ cannot determine whether Bob is talking to the same person as before, or not. At most, $Lighthouse_{Bob}$ knows that Bob is communicating with various people registered at the same Lighthouse.

Note that if the two Lighthouses involved in the communication collude, then the connection is no longer anonymous. Likewise, $Lighthouse_{Bob}$ can collude with the lookup service to determine Alice's identity. However, we assume that this is not trivial. Ideally, a large number of Lighthouses spread over different domains will be available making it harder for two particular Lighthouses to collude. Further, recall that certificate authorities will vouch for the trustworthiness of a Lighthouse, and hence a paranoid user could pick a Lighthouse with stronger assurances (higher in the hierarchy). Again, we are distributing the trust in Mist, rather than centering it at one particular place.

4. Implementation

We are incorporating Mist into the Gaia OS [11]. We implement the Mist Hierarchy as an overlay network over TCP/IP. We implement Mist Routers, Portals, and Lighthouses as CORBA components to facilitate their integration into Gaia's infrastructure. Current CORBA implementations are heavyweight and may not be appropriate for routing packets. This drawback is a serious obstacle for the wide-scale deployment of Mist Routers. Therefore, we are experimenting with the Universally Interoperable Core (UIC), which provides a lightweight, high-performance

implementation of basic CORBA services [13]. In fact, UIC also allows for a lightweight implementation on small devices with limited resources. We can envision an environment in which commodity devices can participate in Mist routing. We implement our own CORBA-based lookup service. A single certificate authority that issues certificates to users and Lighthouses is made available.

To demonstrate Mist, we have implemented an instant messaging application in Java, which uses Mist to preserve users' privacy. The application can be run on Java-enabled mobile devices. More details of Mist's design and implementation can be found in sections 3, 4, and 5 of [2].

5. Future Work

We believe our protocol can be enhanced by optimizing the communication. For instance, turning our attention to Figure 1, we can see that all communication from Bob to Alice will first travel up the hierarchy to $\text{Lighthouse}_{\text{Bob}}$, then to $\text{Lighthouse}_{\text{Alice}}$, and finally to Alice. Communication from Alice to Bob is similar. We can see that Alice and Bob pay the penalty of privacy in terms of extra "hops." Ideally we would like to "short circuit" their communication to take the shortest path possible, while still maintaining location privacy and communication anonymity. With respect to the hierarchy, the shortest path will go through a "lowest common ancestor" Mist Router with respect to Alice and Bob. We call this $\text{Lighthouse}_{\text{LCA}}$. We would like to redirect all communication between Alice and Bob to go through $\text{Lighthouse}_{\text{LCA}}$, while maintaining privacy and an anonymous connection. Finding suitable schemes for such an optimization of Mist communication is a subject of future research. However, we have outlined one possible approach in Section 6 of [2].

6. Conclusion

Ubiquitous computing is an emerging research area with great potential. However, without careful consideration for user privacy from the ground up, there is a fair possibility of creating a ubiquitous 'surveillance' system instead. To avoid this undesirable future, we contend that the privacy and anonymity of users in ubiquitous computing environments should be considered seriously and carefully from the very beginning of the system design phase.

In this paper we present a scheme to preserve privacy in ubiquitous computing environments. Our scheme meets the privacy objectives that we set forth, namely, location privacy, connection anonymity, and confidentiality. We describe how Mist Communication can achieve location privacy and connection anonymity through the use of Mist Circuits. The use of session keys in all phases guarantees the confidentiality of Mist messages. Finally, end-to-end communication between users in the Mist can be configured to use any secure communication scheme desired.

As a direct application of our approach to ubiquitous computing environments, we are currently incorporating the Mist system into the Gaia operating system.

7. Acknowledgement

The authors would like to thank Prasad Naldurg for his useful insights.

8. References

[1] Anonymizer, http://www.anonymizer.com

[2] J. Al-Muhtadi, R. Campbell, A. Kapadia, M. D. Mickunas, and S. Yi, "Routing through the Mist: Design and Implementation," UIUC Technical Report UIUCDCS-R-2002-2267, March 2002.

[3] T. Ballardie, P. Francis and J. Crowcroft, "Core Based Trees (CBT), An Architecture for Scalable Inter-Domain Multicast Routing," ACM SIGCOMM, 1993.

[4] Y.K. Dalal and R.M. Metcalfe, "Reverse Path Forwarding of Broadcast Packets," *Communications of the ACM, 21:1040-1048*, December 1978.

[5] Freedom.net, http://www.freedom.net

[6] The Gaia Homepage, http://choices.cs.uiuc.edu/ActiveSpaces/index.html

[7] Y. Guan, C. Li, D. Xuan, R. Bettati, and Wei Zhao, "Preventing Traffic Analysis for Real-Time Communication Networks," *Proceedings of The IEEE Military Communication Conference (MILCOM) '99*, November 1999.

[8] M. Langheinrich, "Privacy by Design – Principles of Privacy-Aware Ubiquitous Systems," *ACM UbiComp 2001*, Atlanta, GA, 2001.

[9] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous Connections and Onion Routing," IEEE Journal on Selected Areas in Communication, Special Issue on Copyright and Privacy Protection, 1998.

[10] M. Reiter and A. D. Rubin, "Crowds: Anonymity for Web Transactions," *ACM Transactions on Information and System Security (TISSEC) Volume 1, Issue 1*, November 1998.

[11] M. Roman and R. Campbell, "GAIA: Enabling Active Spaces," *9th ACM SIGOPS European Workshop*, September 17th-20th, 2000, Kolding, Denmark.

[12] M. Roman, C. Hess, A. Ranganathan, P. Madhavarapu, B. Borthakur, P. Viswanathan, R. Cerqueira, R. Campbell, and M. D. Mickunas, "GaiaOS: An Infrastructure for Active Spaces," Technical Report UIUCDCS-R-2001-2224 UILU-ENG-2001-1731, University of Illinois at Urbana-Champaign, 2001.

[13] M. Roman, F. Kon and R. H. Campbell, "Reflective Middleware: From Your Desk to Your Hand," *IEEE Distributed Systems Online Journal, Special Issue on Reflective Middleware*, July 2001

[14] N. Priyantha, Anit Chakraborty, and Hari Balakrishnan, "The Cricket Location-Support System," *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (ACM MOBICOM)*, Boston, MA, August 2000.

[15] SafeWeb, http://www.safeweb.com

[16] J. Schwartz, "As Big PC Brother Watches, Users Encounter Frustration," *The New York Times*, September 5, 2001.